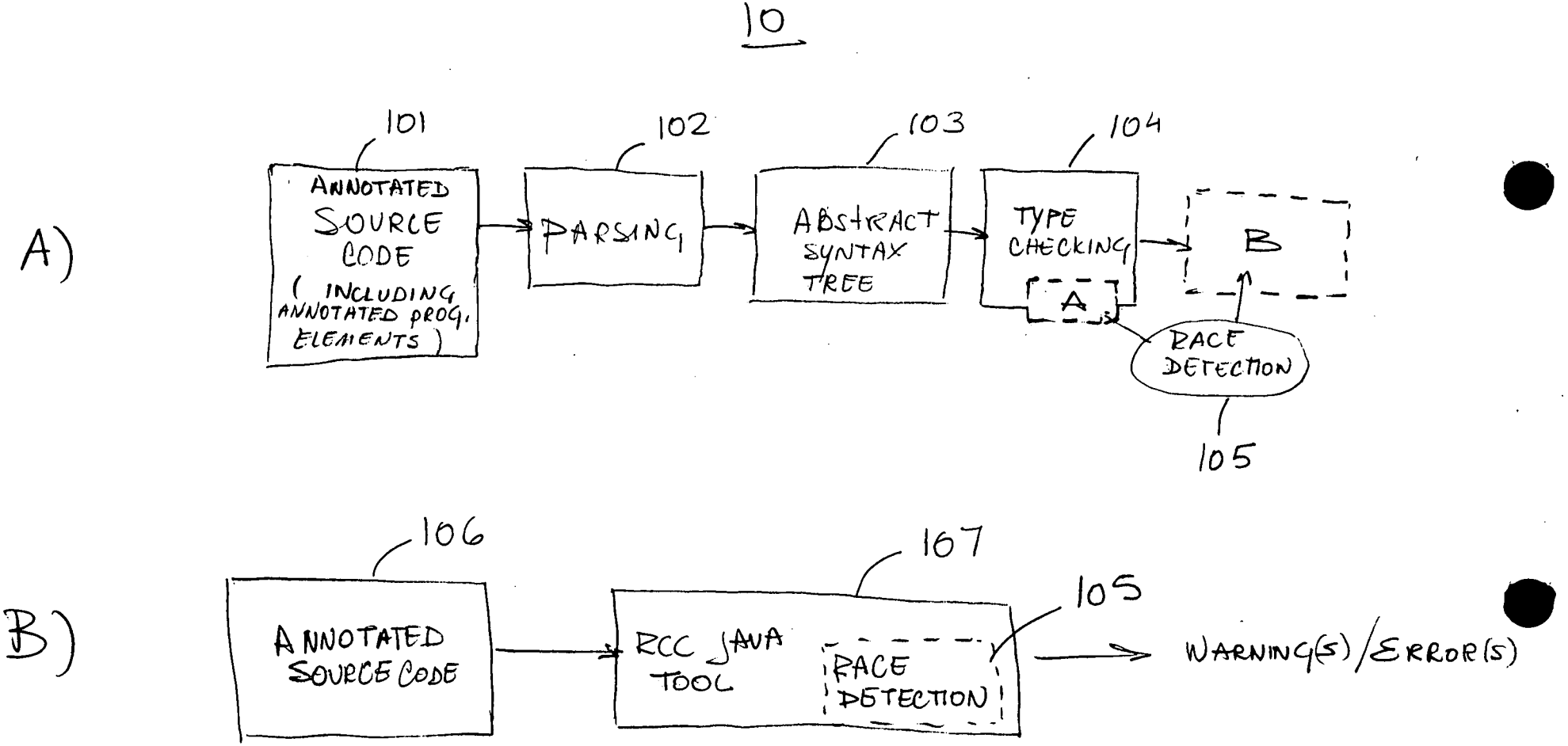
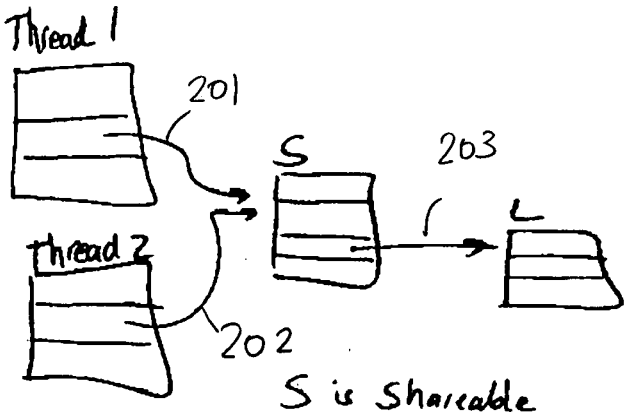

10
A)
101
ANNOTATED SOURCE CODE (INCLUDING ANNOTATED PROG. ELEMENTS)
102
PARSING
103
ABSTRACT SYNTAX TREE
104
TYPE CHECKING
A
B
RACE DETECTION
105
B)
106
ANNOTATED SOURCE CODE
107
RCC JAVA TOOL
RACE DETECTION
105
WARNING(S)/ERROR(S)

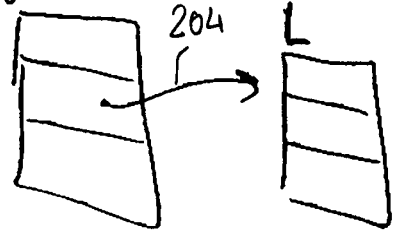
Fig. 1

A)

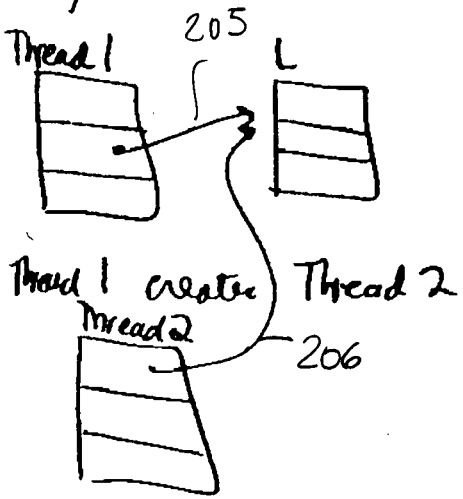


S is Shareable
L is thread-local

B)



C)



- checked at Thread creation points in program text.

Fig. 2

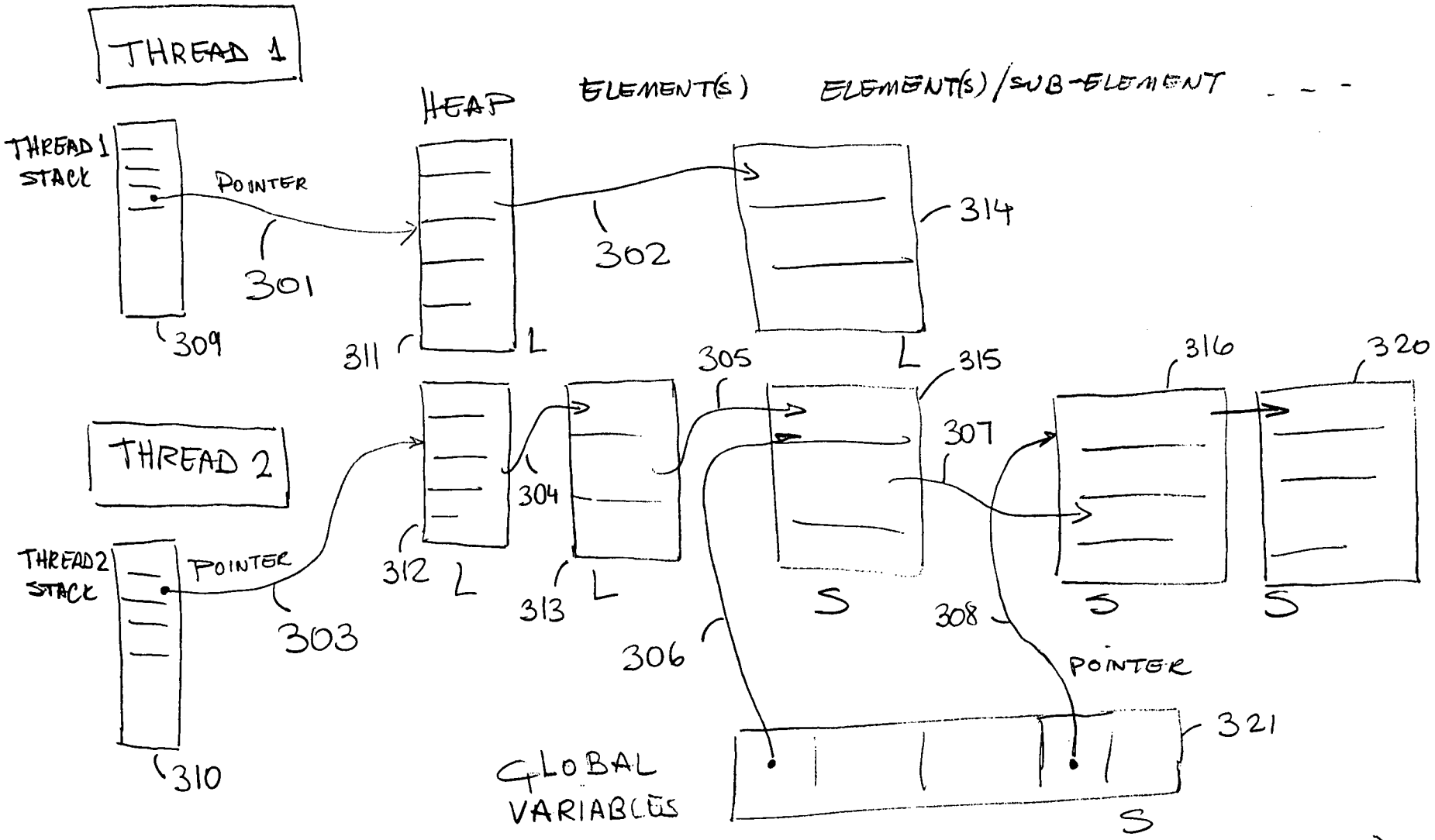

THREAD 1
HEAP
ELEMENT(S)
ELEMENT(S)/SUB-ELEMENT
THREAD 1 STACK
POINTER
301
309
311
L
302
314
305
315
316
320
THREAD 2
307
304
THREAD 2 STACK
POINTER
303
312
313
S
308
306
POINTER
321
GLOBAL VARIABLES
310
L : THREAD-LOCAL
S : THREAD-SHARED
30 - PROG. SOURCE CODE (ANNOTATED)

Fig. 3

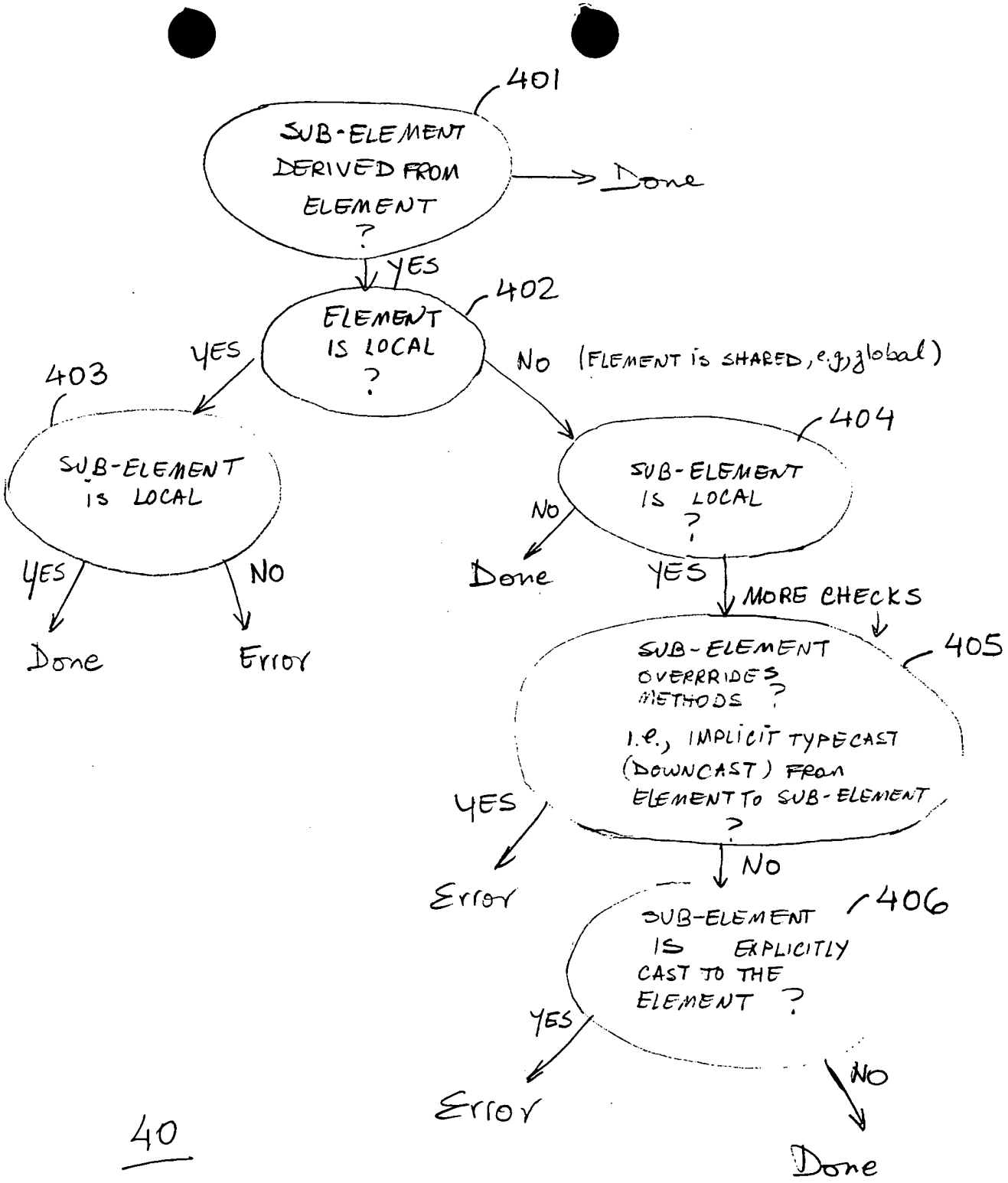

401
SUB-ELEMENT DERIVED FROM ELEMENT ?
Done
YES
402
ELEMENT IS LOCAL ?
YES
No (ELEMENT is SHARED, e.g. global)
403
SUB-ELEMENT IS LOCAL
404
SUB-ELEMENT IS LOCAL ?
No
Done
YES
MORE CHECKS
YES
No
Done
Error
SUB-ELEMENT OVERRIDES METHODS ?
i.e., IMPLICIT TYPECAST (DOWNCAST) FROM ELEMENT TO SUB-ELEMENT ?
405
YES
No
Error
SUB-ELEMENT IS EXPLICITLY CAST TO THE ELEMENT ?
406
YES
No
Error
Done
40

Fig. 4

more cheeks :

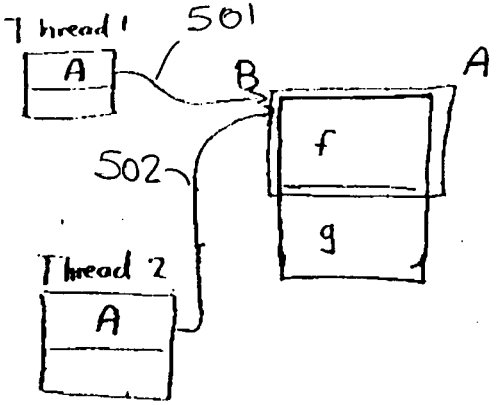
```
thread shared class A {

    Object f;
    void foo() } ... }
}

thread local class B extends A {

    Object g;
    void foo() { ...this.g... }
}
```

this of type B



(typecasts)

ok if no downcasts from A to B

- explicit c is of type A

```
d = (B) c
d.g
```

- implicit

c is of type A (but actually points to a B object)

```
c.foo();
```

Create a B object
Cast it to type A
Pass it between threads
Cast it back to type B
and access fields of B.

Fig. 5

error :

```
thread local   class  A {

     Object f;

}

thread shared  class B  extends  A {

     Object g;

}
```

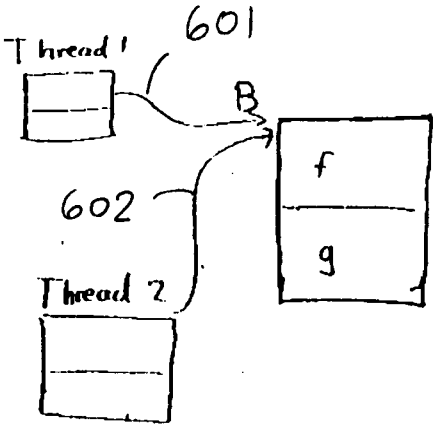


Fig. 6